File Number 1710-36
Form C26-5733-0

IBM Systems Reference Library

IBM 1710 FORTRAN Executive System Specifications

The FORTRAN Executive System provides the user with the ability to direct process control operations with programs written in the FORTRAN language.

Copies of this and other IBM publications can be obtained through IBM Branch Offices. Comments concerning the contents of this publication may be addressed to: IBM, Product Publications Department, San Jose, California

Contents

IBM 1710 FORTRAN Executive System 5
Introduction 5

Disk Loading Program 10
Compiler Output Routines 10

IBM 1710 FORTRAN II-D 12

Executive Control Programs 13
Skeleton Executive Program 13
Master Interrupt Control Program 13
Disk Access Control (DAC) Program 14
Analog Output Control (AOC) Program 15
System Alert Control (SAC) Program 16
Process Schedule Control (PSC) Program 18
Serial Input/Output Control (SIOC) Program 18
Executive Subprograms 20

Preface

The IBM 1710 FORTRAN Executive System provides the 1710 Control System user with the ability to use programs written in FORTRAN language with the Executive programs. The Executive Programming series, first produced as the 1710 Basic Executive and later modified and named the 1710 Executive II, is a group of programs designed to direct and control all facets of process monitoring.

The user-written FORTRAN programs will be compiled using the 1710 FORTRAN II-D Compiler program supplied with the FORTRAN Executive package. Call sequences to the Executive programs are handled by the use of a FORTRAN CALL statement in the user-written program.

To fully understand the material in this publication, the reader should be familiar with the information contained in the following list of IBM publications.

1710 Control System Reference Manual (Form A26-5709)

1710 Additional Special Features and Attached Units (Form A26-5660)

1311 Disk Storage Drive, Model 3 (Form A-26-5650)

1620 Monitor I System Reference Manual (Form C26-5739)

Machine Requirements

The FORTRAN Executive requires that the following features be installed on the user's 1710 Control System with the 1711 Data Converter, Model 2.

1. IBM 1311 Disk Storage Drive, Model 3
2. Indirect Addressing
3. Automatic Divide
4. Basic Interrupt
5. Seek Complete Interrupt
6. Analog Output Setup Interrupt (optional)
7. Serial Input/Output Channel (optional)

Introduction

The preparation of programs for computer control of process operations is simplified by the 1710 FORTRAN Executive System. FORTRAN is a problem oriented programming system. Using FORTRAN, the programmer can describe process relationships in a format close to that of mathematical equations. The Executive Control programs provide the user with an efficient method of controlling his process. The combination of the versatile FORTRAN language and the Executive Control programs makes a powerful tool that provides the user with a means of making the most effective use of his IBM 1710 Control System. The FORTRAN Executive can be logically divided into the three parts listed below:

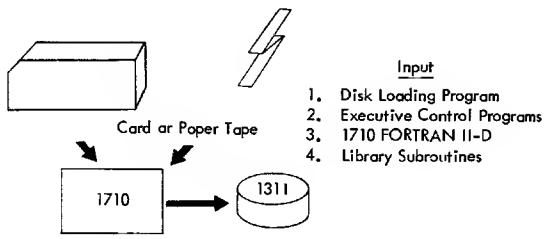
1. FORTRAN II-D. This is a compiler program, i.e., it is a program that accepts user-written statements as input and produces compiled machine language instructions as output. FORTRAN II-D is used in the first step toward "organizing" a process-control program to operate with the FORTRAN Executive System.
2. Disk Loading Program. This program moves programs to and from disk storage and core storage. Furthermore, it creates (and deletes) entries for various communications areas that are used (a) by the Disk Loading program at *load* time, and (b) by the Executive Control programs during *execution* of the user's program. The Disk Loading program is used in the second step of organizing a process-control program.
3. Executive Control Programs. These programs (and subprograms) supervise the transfer of programs and data between disk storage and core storage, read input data from the process, control output signals, and handle all error conditions during execution of the user's programs. The user can call for the Executive programs at any time by using a CALL statement in his program.

Figure 1 shows the flow of control between the user's programs and the programs and routines of the 1710 FORTRAN Executive. All the programs are stored on disk storage so that during control of the process any of the user's programs or any of the Executive programs may be transferred from disk storage to core storage for execution.

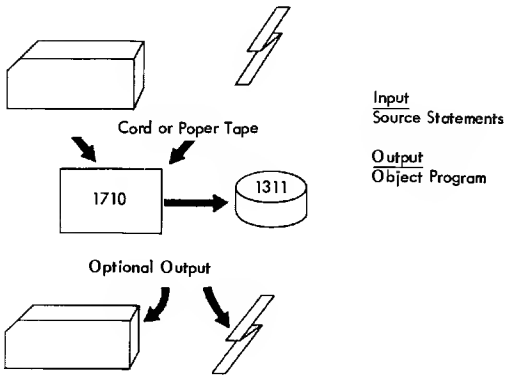
General Operation

The implementation of the FORTRAN Executive System consists of four phases.

Phase 1. Phase 1 consists of loading the IBM supplied programs to disk storage. The programs are self-loading and are supplied in card or paper tape form. Through the use of a control statement, certain parameters for the Executive programs can be assigned at this time.



Phase 2. Phase 2 consists of using the 1710 FORTRAN II-D program to compile user-written programs. Input may be in the form of punched cards or paper tape, and output may be in cards or paper tape. Compilation is done with the interrupt feature disabled.



After compiling, the object program is in *relocatable* form, i.e., the addresses of the instructions must still be modified relative to the program starting address. The modification must take place before the object program can be executed.

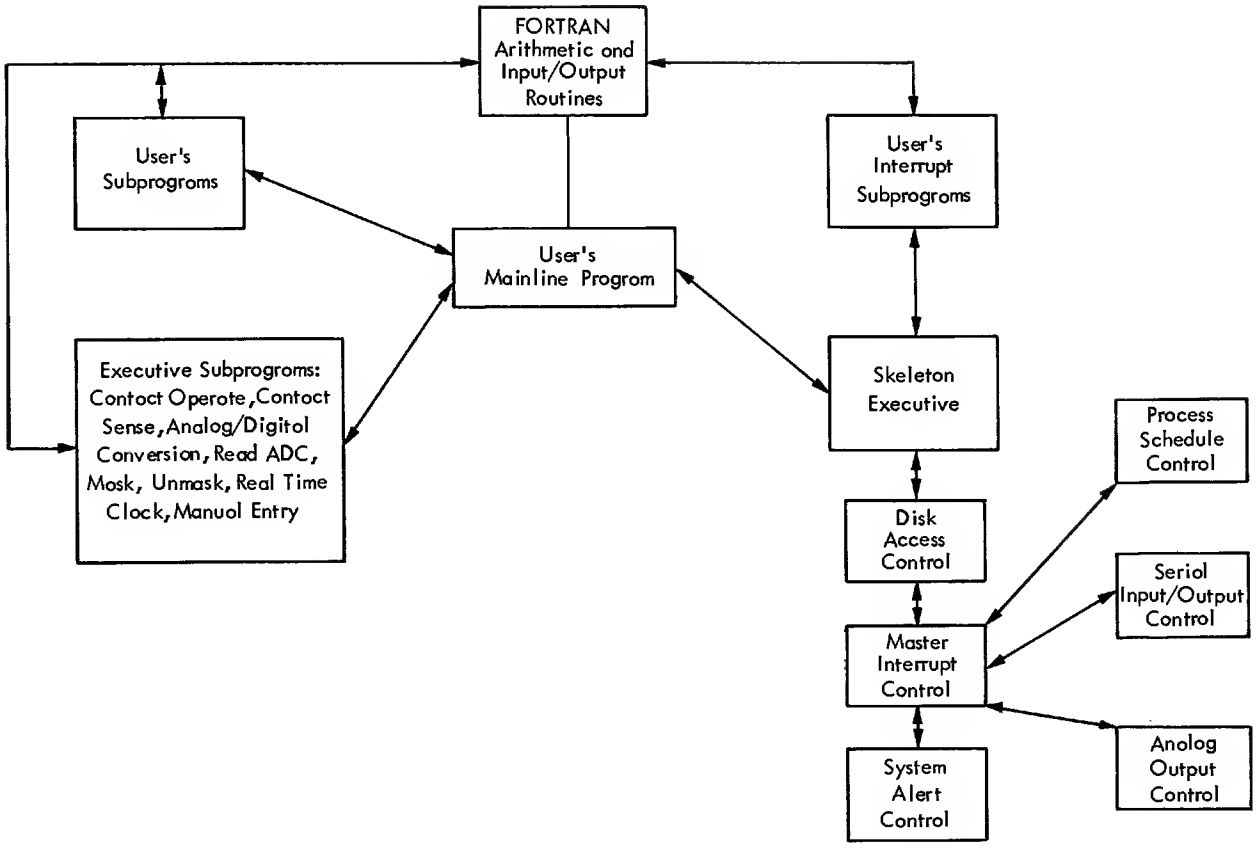
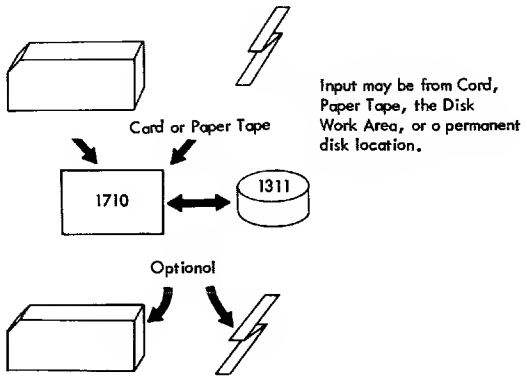


Figure 1. Flow Chart of Program Control

Phase 3. Phase 3 consists of using the Disk Loading program to load the object programs (output from FORTRAN II-D) to disk storage. The object program may be loaded from cards, paper tape, or from disk storage itself. The program being loaded may be loaded to disk storage as it is (in relocatable form) or it may be changed to *core image* form (with absolute addresses) so that when the program is loaded to core storage it can be executed immediately.

For permanent storage, programs are combined into groups called "core loads." A core load consists of the core image form of a main program and the subprograms that it utilizes. Core loads are always kept on disk storage and the relocatable form of the programs may also reside on disk.

When making up core loads, the Disk Loading program begins its task by changing the main program to core image form. Then it loads this program to disk storage, makes entries for various maps, changes the subprograms to core image form, loads them to disk storage, and again makes map entries. Maps are simply records of data containing addresses and identification information (see COMMUNICATIONS AREAS).

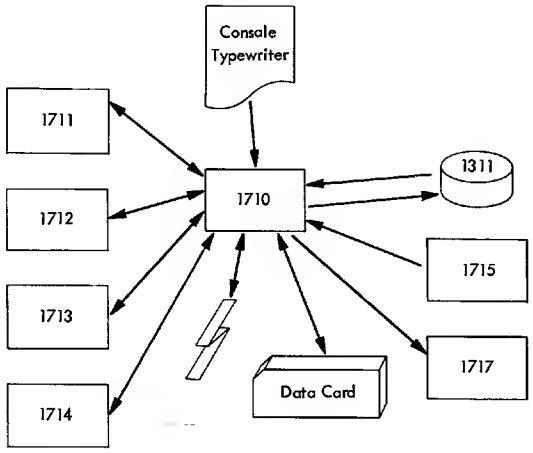


Phase 3 is completed when all programs and subprograms have been loaded to disk storage. The programs are in executable form, so that when they are called from disk storage to be executed, no time will be lost adjusting addresses.

Phase 4. This is the execute phase. The operator first calls in the Skeleton Executive loading routine. This routine loads the:

Multiply and Add tables
FORTRAN Arithmetic and Input/Output routines
Skeleton Executive program
Core portion of the Disk Access Control program

The user then specifies the desired mainline program, and the core load is loaded into core storage. After the core load is loaded, a branch occurs to the starting address of the user's main program. From this point the 1710 Control System is capable of controlling the process. The degree of control the computer maintains over the process is dependent only on the machine configuration and the user's programs.



Communications Areas

There are several areas of disk storage and core storage that must be reserved for use as communications areas by the user's programs and the programs that make up the FORTRAN Executive System. The disk storage areas must be assigned before loading any user-written programs (see DISK LOADING PROGRAM).

CORE STORAGE

1. FORTRAN COMMON Area
2. Program Linkage Area

FORTRAN COMMON Area. This area is assigned by the user through the use of COMMON and DIMENSION statements in the main program and subprograms. This area will contain tables and any other information deemed necessary by the user.

Program Linkage Area. This area contains the information needed by the main program (in core) to call a subprogram in from disk storage. The linkages are put together by the Disk Loading program when main programs are loaded to disk storage (to form a core load). Each linkage contains the disk sector address, sector count, and the core address where the subprogram is to be loaded. The program linkage area is entered by a branch instruction from the main program. When the program linkage area is entered, another branch is executed to a routine that reads in the subprogram desired. After the routine is read into core storage, a branch instruction that is in the read-in routine transfers control to the subprogram.

DISK STORAGE

1. Core Load Map
2. Disk Identification Map
3. Interrupt Subprogram Identification Map
4. Equivalence table

Core Load Map (CLM). The CLM is used by the Executive programs to determine how each core load is to be handled. The Core Load Map is a series of records that contain address and status information. This map is created by the Disk Loading program when organizing a core load. One entry (record) for the Core Load Map is created for each core load. Each record is divided into three tables (see Figure 2).

1. Core Load table
2. Interrupt Subprogram Status table
3. Interrupt Priority table

Core Load Table. This part of a record in the Core Load Map contains core load identification and address information. It consists of 28 core locations apportioned in the following manner:

1. Three-digit identifier of the core load related to this record.
2. Three-digit identifier of the next core load (the core load the user wants to have executed at the completion of the current core load).
3. Three-digit identifier of exception procedure core load (see SYSTEM ALERT CONTROL PROGRAM).
4. Three-digit identifier of the restart procedure core load (see SYSTEM ALERT CONTROL PROGRAM).
5. Six-digit address of the current mainline program.
6. Three-digit sector count of the current mainline program.
7. Five-digit core storage address which is the beginning core storage address and starting address of the current mainline program.
8. One-digit Alert Procedure indicator. This indicator is interrogated by the System Alert Control program to determine how the user wants a

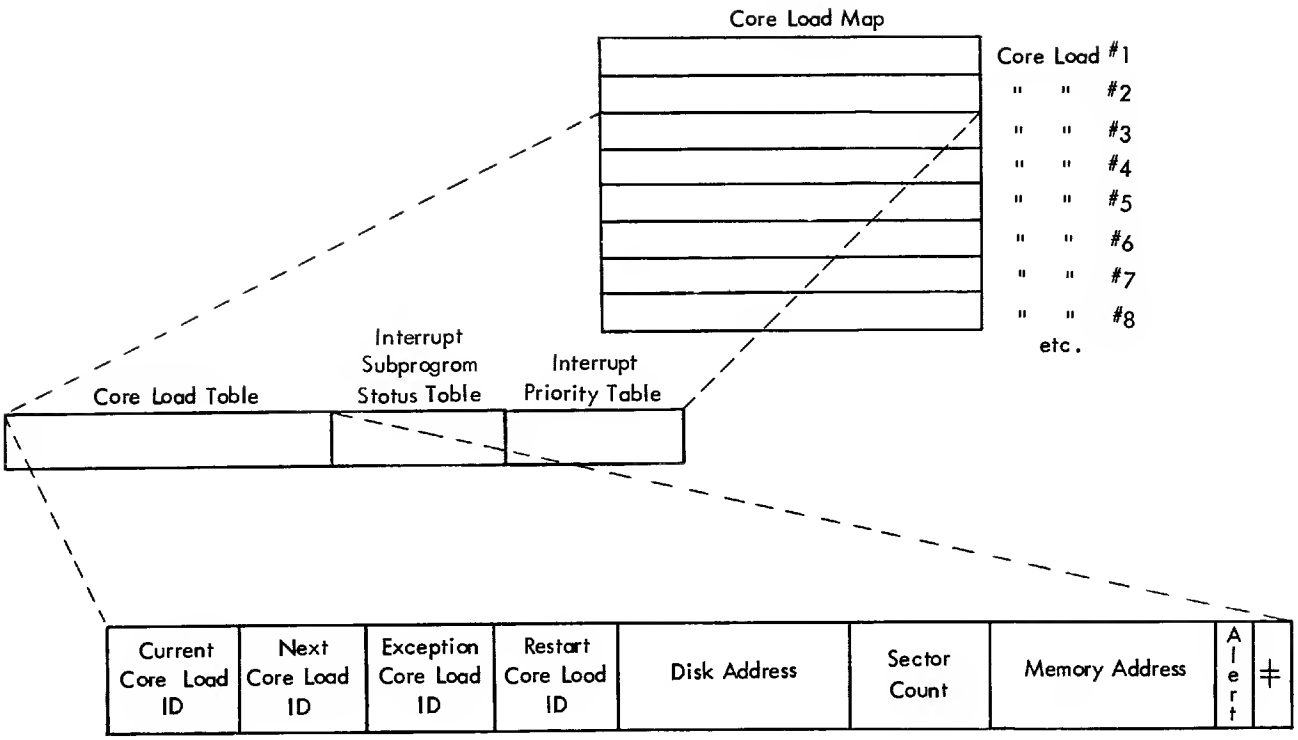


Figure 2. Core Load Map

particular error condition to be handled. The user has three choices:

1 – halt the program
2 – record the error, but do not halt
3 – branch to exception procedure program

9. Record mark.

The map entry for a particular mainline program is loaded to core storage with that program. It is referred to as the "current" record in the Core Load Map.

Interrupt Priority Table. As its name suggests, this part of the core load map determines the order in which the various interrupt indicators are to be interrogated. It consists of a two-digit field for each interrupt. The number in the field specifies the order directly, e.g., 48 in field three means that interrupt indicator number 48 is to be the third indicator tested. The interrupt indicator numbers are assigned by using the INTPR control statement.

Interrupt Subprogram Identification Map (ISIM). The ISIM consists of a series of 16-digit records (see Figure 3). There is a record for each interrupt subprogram. Each record is composed of the following fields:

1. Six-digit disk address of the subprogram.
2. Three-digit sector count of the subprogram.
3. Five-digit core storage address where the subprogram is to be placed. This is also the starting address of the subprogram.
4. One-digit status control code.
5. Record mark.

The ISIM is updated each time a core load is transferred to core storage for execution.

If an interrupt occurs and the interrupt subprogram associated with it is not to be executed immediately, the interrupt is recorded by the Master Interrupt Control program which places a flag over the record mark in the corresponding ISIM record. See PROCESS SCHEDULE CONTROL PROGRAM for a description of how recorded interrupts are serviced.

The address information in the ISIM is utilized by the Disk Access Control (DAC) program when certain call sequences are given. The ISIM allows the user to refer to subprograms without knowing specific addresses. This is a significant point because it permits the redistribution of subprograms and data on disk storage without the necessity of reassembling the calling programs.

Disk Identification Map (DIM). The DIM consists of entries that are 20 positions in length. Each entry specifies the disk address and the length (in sectors) of a program that is located on the disk. It also contains core addresses for the program (see Figure 4). A map

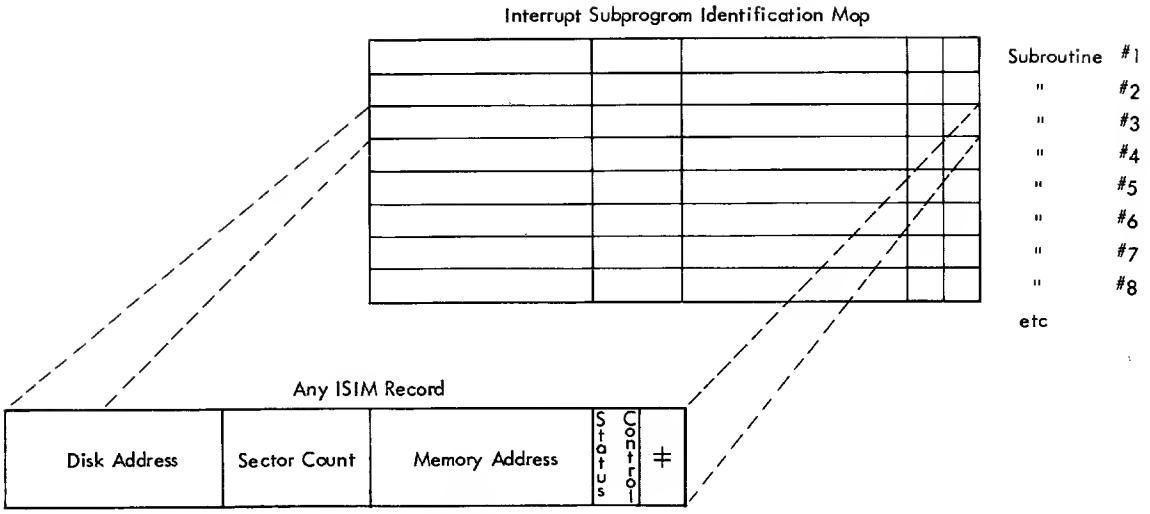


Figure 3. Interrupt Subprogram Identification Map

entry is created when a program is loaded to disk storage.

The Disk Loading program will assign the Disk Identification Map entry number in consecutive sequence starting with the lowest available entry. When a user wishes to change or replace a program, the Disk Loading program will change the map entry to reflect the change, if any, in disk storage. If the map entry is deleted, the position of this entry becomes available for future use. All changes to the DIM will result in automatic updating of the Equivalence table.

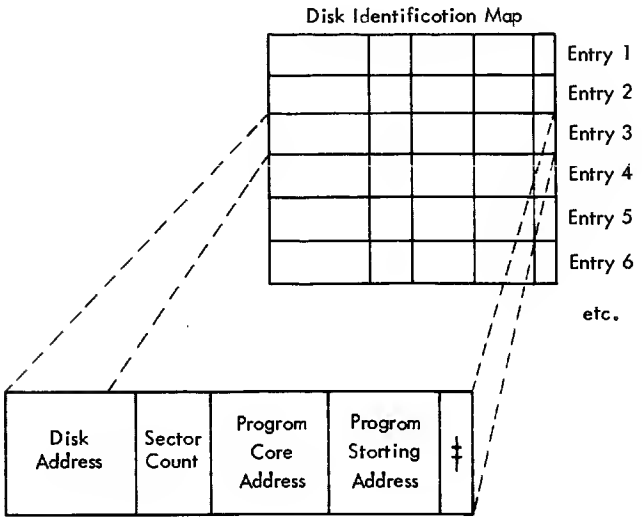


Figure 4. Disk Identification Map

Equivalence Table. The Equivalence table contains the alphabetic name and the DIM entry number of each program that has been assigned a name by the user. A name is not necessary unless the program is to be referenced using the name. In many cases only a DIM number is needed. The format of the Equivalence table is shown with an example below:

$$\underbrace{\bar{4}142434400000}\underbrace{\bar{0}234}$$

Program Name — DIM Entry Number

This example shows a program named ABCD as being equivalent to DIM entry number $\bar{0}234$.

Sequential Program Table. This table contains the four-digit DIM entry number for every program and data area. In addition to the DIM entry, a special four-digit field is used to denote the available storage area between programs. This four-digit field contains a 9 in the high-order position for identification. The three low-order positions denote the number of available consecutive sectors; for example,

$$\bar{0}4189021\neq$$

indicates that 21 sectors are unused by programs or data following the program identified with DIM entry $\bar{0}418$.

Disk Loading Program

The purpose of the Disk Loading program (DLP) is to efficiently deal with all disk loading and maintenance. The DLP is made up of a group of routines that reside on disk and are called into core storage and executed by use of a control statement. The type of control statement determines which operation the user wants the Disk Loading program to perform. The communications areas that are formed and acted upon by the DLP are listed below.

DLP Communications Areas

Core Load Map (CLM). An entry in the CLM is made for each core load. The Core Load Map is created by the Disk Loading program from (1) data the user supplies in a DLOAD or DREPL control statement, and (2) from tables and other maps.

Interrupt Subprogram Identification Map (ISIM). An entry is created in the ISIM for each subprogram that is identified as an interrupt subprogram at load time.

Disk Identification Map (DIM). An entry is automatically placed in the DIM each time a new program is loaded to the disk. The Disk Identification Map contains entries for all portions of the disk occupied by data or programs.

Equivalence Table. An entry to this table is created by the DLP when a new program is loaded to disk. The table contains the user-assigned name of the program and the DIM entry number. In this way a cross-reference is established whereby programs can be referred to by the name or DIM entry number.

Sequential Program Table. The order of the programs is contained in this table. Whenever a program is added or deleted, the Sequential Program table is updated to reflect the change. The length of the table is 199 sectors, unless changed by a control statement.

DLP Control Statements

Control statements are used to direct the Disk Loading program to accomplish a desired operation. A statement can be used to define program type, assign user-specified addresses of core storage and disk storage, load programs, and make entries for communications areas, etc. Before loading any user-written programs, an option is available for the user to reassign reserved areas of disk storage for various work and communications areas. A description of the use of the control statements follows:

DLOAD. The DLOAD statement assigns programs to disk storage. It causes the DLP to update the system communications areas and to modify the program being loaded into core image form (absolute addresses), if it is in relocatable form.

DELET. The DELET statement deletes programs and associated DIM entries (and ISIM entries, if applicable) and Equivalence table entries.

DREPL. The DREPL statement replaces a program in disk storage with an updated, changed, or new program.

DDUMP. This statement transfers data or programs from disk storage to cards, paper tape, or the typewriter.

INTCR. The INTCR statement identifies interrupt subprograms that are to be loaded as part of the main core load.

INTPR. The INTPR statement identifies the sequence in which process interrupts are to be checked, i.e., it assigns the priority.

LOCAL. The LOCAL statement identifies user subprograms as "load-on-call" subprograms. A subprogram called in a main program will be loaded as part of the main core load if not identified as a load-on-call subprogram. If defined as a load-on-call subprogram, it is not loaded with the core load but remains on disk storage until called.

DFINE. The DFINE statement allows the user to enlarge, shorten, or relocate the disk storage areas assigned for work cylinders, DIM, ISIM, and the Equivalence table. If used, the DFINE statement should precede loading of any user's program to disk storage.

Compiler Output Routines

The DLP also contains the output routines for the FORTRAN II-D program. At compile time the user indicates the type of output desired, and the FORTRAN

compiler calls the DLP routine when compilation is complete. At this time the object program is in the "work area" in disk storage. Depending upon the option(s) chosen, DLP will load the object program to a permanent area of disk storage and/or punch the object program into cards or paper tape. Object programs punched or loaded in this manner are in the relocatable form. That is, all object programs are created with a beginning address of 00000. The program instruction addresses must be modified before the program can be executed. The modification is done by the DLP when core loads are being formed. In this way, a program is compiled *once*, then it can be modified n-number of times so it can be executed from n-number of locations in core storage. When a program has been modified, it is in core-image form. Up to nine core-image versions of a program may be on disk storage.

The 1710 FORTRAN II-D program is a compiler program; it reads user-written FORTRAN statements in the form of punched cards or paper tape, and compiles an object program. It is similar to the 1620 FORTRAN II-D program, and, with few exceptions, all the statements that are available for use with the 1620 version apply to the 1710 version.

The user may write process interrupt subprograms in the FORTRAN II-D language. The FORTRAN arithmetic and input/output work areas will be stored on the disk in a "save" area when a process interrupt occurs. Then the interrupt subprogram is transferred to core storage and executed (it may also be necessary to save the core area that the interrupt subprogram occupies while it is being executed — see EXCHANGE OPERATION). During execution of the interrupt subprogram, the FORTRAN arithmetic and input/output subroutines may be used. But, before operation of the original program (the program that was being executed at the time of the interrupt) can be resumed, the FORTRAN arithmetic and input/output work areas are restored because the original program could have been in the middle of an operation with partially calculated data in the FORTRAN work areas. When the work areas are restored, control is returned to the original program.

Statement Exceptions

The following FORTRAN statements that are acceptable in 1620 FORTRAN II-D are *not* allowed in the 1710 FORTRAN II-D program.

```
STOP
CALL EXIT
CALL LINK
```

INTERRUPT SUBPROGRAM CONTROL STATEMENT

Source statements for interrupt subprograms must be identified with an INTER control statement. This control statement serves as notification to the FORTRAN program to compile a branch instruction to the Master Interrupt Control program for RETURN statements.

When the RETURN statement instruction for an interrupt subprogram is executed, a branch occurs to the Master Interrupt Control (MIC) program to return control to the mainline program. This allows waiting interrupts to be serviced as soon as possible.

CALL EXEC Statements

The user may call the Executive Control programs into operation at any time by using a CALL statement in his programs. The statement may be used in both main programs and subprograms. The general form of the statement is as follows:

CALL EXEC (IDENT, P_1, P_2, . . . P_n)

The parameter list is variable in length, but IDENT is always the control program identification. For detailed descriptions of the CALL statements and their parameter lists refer to CALL STATEMENT under each specific control program.

The CALL EXEC statement, when executed, transfers control to the proper Executive Control program.

FORTRAN Disk Work Area

The FORTRAN compiler program uses a "work area" of disk storage when compiling programs. This work area is assigned from disk cylinder 00 through cylinder 23. The area can be changed by using a DEFINE control statement and specifying the desired cylinders. The DEFINE control statement must be entered before any data has been loaded to the disk pack being used.

At the end of compilation, the object program will reside on disk even if the user has obtained card or paper-tape output from the compiler. The object program in the work area is in relocatable form and (1) can be moved, in this form, to a permanent area of disk storage, or (2) can be changed to core-image form and assigned a permanent location on disk. The move operations would be performed through the use of DLOAD or DREPL control statements to the Disk Loading program.

Skeleton Executive Program

The Skeleton Executive program handles all requests for entry to and exit from the user's programs and the Executive program. The Skeleton Executive saves the FORTRAN arithmetic and input/output work area when an interrupt occurs or when a subprogram is called in to core storage from disk storage.

Whenever a call to one of the Executive programs is executed, the Skeleton Executive program transfers the parameters of the CALL statement and temporarily stores them in the CALL area of the Skeleton Executive. The parameters are then analyzed, the required area of core storage is transferred to the "save area" of disk storage, and the proper Executive program is transferred to core storage for execution.

The Skeleton Executive program is available in two versions. The first version takes care of the aforementioned tasks and always resides in core storage during execution of the user's programs. The second version also continually resides in core storage, but requires a slightly larger area as it contains the routines required for operation of an SIOC output printer (see SERIAL INPUT/OUTPUT CONTROL PROGRAM).

Operation

When an interrupt occurs, the Skeleton Executive takes control and:

1. Interrogates the process interrupt indicators associated with the interrupt-subprograms-in-core. If one of these indicators is on, control is transferred to the interrupt-subprogram-in-core.
2. If an indicator associated with the interrupt-subprograms-in-core is *not* on, the SIOC interrupt is interrogated. If the SIOC interrupt is on and is associated with an output printer, control is transferred to the SIOC output printer routine (version 2 of the Skeleton Executive is assumed in this case).
3. If none of the previously mentioned indicators are on, the Skeleton Executive saves the required area of core storage (transfers the area to disk storage). Then the Master Interrupt Control program is called into core and executed.

Master Interrupt Control Program

The MIC program has one important function: it serves as the interrupt identification routine. Specifically, it

1. Determines which interrupt(s) occurred.
2. Services interrupts in the user specified sequence.
3. Records interrupts for servicing at a later time.
4. Returns control to the Skeleton Executive and thus to the mainline program when no interrupts remain to be serviced (see Figure 1).

The return path from interrupt subprograms to the mainline program is through the MIC. The means of exit from an interrupt subprogram must always be a RETURN statement.

Operation of MIC

When the computer is in the interruptible mode and an interrupt occurs, one of three courses of action will result:

1. The interrupt is associated with an interrupt-subprogram-in-core, chosen by the user. In this case, a branch to the interrupt-subprogram-in-core occurs and the subprogram is executed immediately.
2. The interrupt is associated with an SIOC output printer, and, since this program is in core, a branch to and execution of the SIOC output printer routine occurs.
3. The interrupt is (a) created by a process interrupt associated with a subprogram that resides on disk storage or, (b) the interrupt is associated with an internal interrupt. Since the interrupt was not associated with the SIOC output printer program nor the interrupt-subprogram-in-core, the Master Interrupt Control program is called into core storage.

Indicators for process interrupts are tested in a sequence that is prescribed by the user in the Interrupt Priority table of the Core Load Map. If an indicator is on, MIC checks the status control digit in the ISIM to determine whether the interrupt is to be serviced at once or recorded for servicing at a later time. If the

interrupt is to be serviced immediately, MIC ascertains whether or not the interrupt subprogram or control program that is needed to service the interrupt is in core storage. If it is in core storage, the FORTRAN arithmetic and input/output work areas are stored on the disk and control is transferred to that interrupt subprogram or control program. If it is not in core storage, the Disk Access Control program is called to preserve core storage (the FORTRAN subprogram area and the area presently occupied that the needed subprogram will fit into) and to bring the needed subprogram in from disk storage.

Interrupt Indicators Interrogation

Interrupts are classed as internal or external depending upon their origin. Those that come from within the control system (computer) are internal; those that originate within the process are external.

INTERNAL

The internal interrupts are listed below with the name of the program that MIC branches to when the interrupt occurs.

Interrupt	MIC branches to
Any Check	System Alert Control
Seek Complete	Disk Access Control
Any SIOC	Serial Input/Output Control
Multiplex Complete	Analog/Digital Control
CE Interrupt	CE Interrupt Subroutine
Operator Entry	Program specified by user
Analog Output Setup	Analog Output Control
One Minute	Program specified by user
One Hour	Program specified by user

EXTERNAL

There are twelve process interrupt indicators available for assignment by the user. When any one of them is on, MIC checks the status control digit in the ISIM and either records the interrupt or transfers control to the interrupt subprogram designated to service that particular interrupt.

Disk Access Control (DAC) Program

The DAC program is used to coordinate the large storage capacity of the disk drive and the high-speed computing facility of core storage. It has three specific tasks:

1. Read from disk storage.
2. Write on disk storage.
3. Exchange data between core storage and disk storage.

A request for disk activity causes at least one of three operations: seek, read, or write as described below.

Seek. When a seek operation is called for (by means of a FORTRAN FIND statement), the disk access arm is positioned over a cylinder.

Read. When a read operation is called for (by means of a FORTRAN FETCH statement), the position of the access arm is checked and one of the following sequences of events takes place:

1. If it is at the required position, the requested information is read into core storage.
2. If the access arm is not at the required position, a seek is initiated, and, when completed, the requested information is read into core storage. (The seek time is unavailable for computation.)

Write. The write operation is similar to the read operation, except that data transfer is from core storage to disk storage. The write operation is called for by the FORTRAN RECORD statement.

Exchange Operation

An "exchange" is a procedure whereby an interrupt program or an Executive program on disk storage can temporarily replace data in core storage while saving the replaced data in a disk buffer area. It is used when a requested program or subprogram is not in core storage. The sequence of operations is as follows:

1. The Skeleton Executive program determines that the requested program is not in core storage.
2. The Skeleton Executive turns control over to the Disk Access Control program which transfers to a disk buffer area the core storage data that is to be replaced.
3. The Disk Access Control program (DAC) reads in the requested interrupt subprogram or Executive program.
4. After execution of the requested subprogram or program, DAC transfers the data saved in the disk buffer area back to its original location in core storage.

If an interrupt subprogram, which is in core storage as a result of an exchange, requests either the Analog/Digital Control program or the Analog Output Control program, another exchange is performed. This time, however, a different buffer area must be used because the original area still contains the data that was removed from core storage to make room for the interrupt subprogram. For a second exchange operation, the Disk Access Control program proceeds as follows:

1. Stores the interrupt subprogram in the second disk buffer area.
2. Reads in the requested Executive program.

3. Returns the interrupt subprogram to core storage after execution of the Executive program.

No more than two buffer areas are ever used by the Disk Access Control program for the purpose of exchanging as a result of an interrupt.

Additional Modules

Additional modules may be used for storage of programs and data, but all portions of the FORTRAN Executive System must reside on the first module.

Analog Output Control (AOC) Program

The purpose of the AOC program is to select and adjust the various set-point positioners (SPP) within the user's process. In doing this, the program allows the user to specify different rates of adjustment so that set-point movements can be synchronized.

Analog Output Logic

The analog output area of a compiler-controlled process consists mainly of controlling instruments operated by set-point positioners. Each SPP is under control of the 1710 System, which determines the frequency and extent of adjustment through the use of the analog output timer. The analog output timer is composed chiefly of a continuously running motor that completes a cycle every 3.6 seconds. This cycle is divided into two parts: a 0.7-second setup period, and a 2.9-second action period. The setup time is used to select the points needing adjustment; the action time is used to actually perform the adjustment.

The adjustment of the SPPs can be accomplished by either a 2.5-second signal (slew) or a 0.5-second signal (trim). Depending upon the desired setting, an SPP may require several slews and trims to bring it into proper adjustment. A slew or trim adjustment can be made only once during each 3.6 second cycle; however, this one adjustment services all the SPPs that were selected during the setup time.

Analog Output Setup Interrupt

This signal interrupts the mainline program at the beginning of the setup portion of the analog output cycle. It ensures that the AOC program has the maximum setup time for selecting the analog output points to be adjusted and for executing a slew or trim operation. This interrupt is initiated every 3.6 seconds whether or not analog output is addressed. The regularity of the interrupt makes it a timing device that can be used for other program functions for which a 3.6-second cycle is wanted. If the Analog Output Setup interrupt is not available, the Multiplexer Complete interrupt can be used.

CALL Statement

Whenever an SPP is to be analyzed and/or adjusted, the programmer inserts a CALL statement in his program in the following form.

CALL EXEC (IDENT, P_2, P_3)

IDENT — 040FF, where FF is the frequency 1-99.
P_2 — 00PPP, where PPP is the SPP address.
P_3 — 00SST, where SS is the number of slews to be executed and T is twice the number of trims to be executed.

Frequency. This indicates how often the SPP will be serviced. The digit in the parameter entry ranges from 1 to 99, where 1 calls for service every 3.6-second cycle, 4 calls for service every fourth 3.6-second cycle, etc. Thus, the higher the number, the lower the frequency of adjustment.

SPP Address. The terminal address for upscale movement of the SPP being operated upon. This address plus one is the terminal address for down-scale movement.

Operation

The AOC program may be logically divided into two operating phases. The first is the initializing phase which sets up the conditions for selecting and adjusting the SPPs; the second is the service phase which actually selects the SPPs and starts the slew and trim operations.

Initializing Phase. When a CALL statement to AOC is executed, the initializing phase is entered and the following events occur:

1. The CALL statement parameters are stored in the applicable record in the Analog Output table portion of the AOC program.
2. If a slew is required, a slew/trim indicator is set to slew. The slew/trim indicator is used by the AOC program to determine which type of operation (slew or trim) is to be performed next.

Since the interval between the setup times is 3.6 seconds, many SPPs can be activated before the service phase is entered.

Service Phase. This phase begins when the Analog Output Setup interrupt occurs. The operations performed in this phase are listed below:

1. The slew/trim program indicator is interrogated to determine if a slew operation has been requested for any SPP. If it has, the addresses are selected for each SPP record that requires slewing.
2. After all SPPs which require slewing have been selected, the Analog Output Setup indicator is tested and, if it is still on, the slew operation is readied.

NOTE: This AO Setup indicator (28) is *not* the same as the AO Setup Interrupt indicator (41). Indicator 41 is turned off when the interrupt indicator is tested, but indicator 28 remains on until the 0.7-second setup time has elapsed.

3. When the slew portion of the output cycle is reached, the slew is performed, and all selected SPPs are adjusted. No further adjustments are made until the next 3.6-second cycle. When no more slews are needed, the slew/trim program indicator is set to trim; trims are then performed in the same manner as were slews.

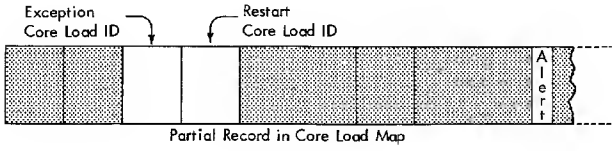
System Alert Control (SAC) Program

The SAC program takes control of the 1710 whenever an error condition is detected. SAC determines which error(s) is present, records each error by type, analyzes the error(s) with respect to operating conditions, and decides which of the following correction procedures is to follow:

1. Restart, using the program specified by the user in the current record of the Core Load Map.
2. Branch to the exception program specified by the user in the current record of the Core Load Map.
3. Record the error, but do not halt.
4. Halt.

Corrective Procedure

To a great extent, the corrective procedures previously mentioned are user-controlled by certain fields in the Core Load Map shown below.



However, some actions taken by the SAC program are mandatory due to the type of error. Figure 5 shows the logic of corrective procedure selection.

Restart Procedures

When SAC branches to the restart procedure, a chain of events is initiated which ultimately results in a new core load in core storage. New subroutines, data, etc., might be loaded over data that is already in core storage. Therefore, when the user requests this procedure, he must be very careful not to destroy any updated information.

Exception Procedure

The exception procedure operates similarly to the restart procedure. There is one major difference, however. The SAC program, considering the possibility that the user's exception procedure program might consist of some type of error analyzation, provides the following information for interrogation:

1. A two-digit "alert" code which provides the user with diagnostic information concerning the error.
2. A five-digit core storage address of the Interrupt Subprogram Identification Map record, which pertains to the process interrupt subprogram being executed. If a process interrupt subprogram is not being executed, this data will not be provided.
3. A three-digit identifier of the current core load.

Error Conditions

The error conditions that SAC analyzes and records are shown in Table 1.

A counter, used for recording purposes, is associated with each error. The contents of the counters are typed out on the console typewriter when the CE Interrupt switch is depressed. After the typeout, the counters are reset to zero.

The SAC program has the ability to logically disconnect any SIOC unit when an SIOC error is repetitive. In either case, the instruction is re-executed in an attempt to correct the error before the unit is disconnected.

Table 1. Error Conditions

Nome	Indicator Code
1620	
Read Check	06
Write Check	07
MAR Check	08
MBR-E Check	16
MBR-O Check	17
1711	
*Any Check	19
Function Register Check	22
TAS Check	21
Analog Output Check	23
1311	
Address Check	36
Record Length Check/Read Back Check	37
Cylinder Overflow	38
*Any File Check	39
SIOC	
Peripheral Error	6043

*No error count kept

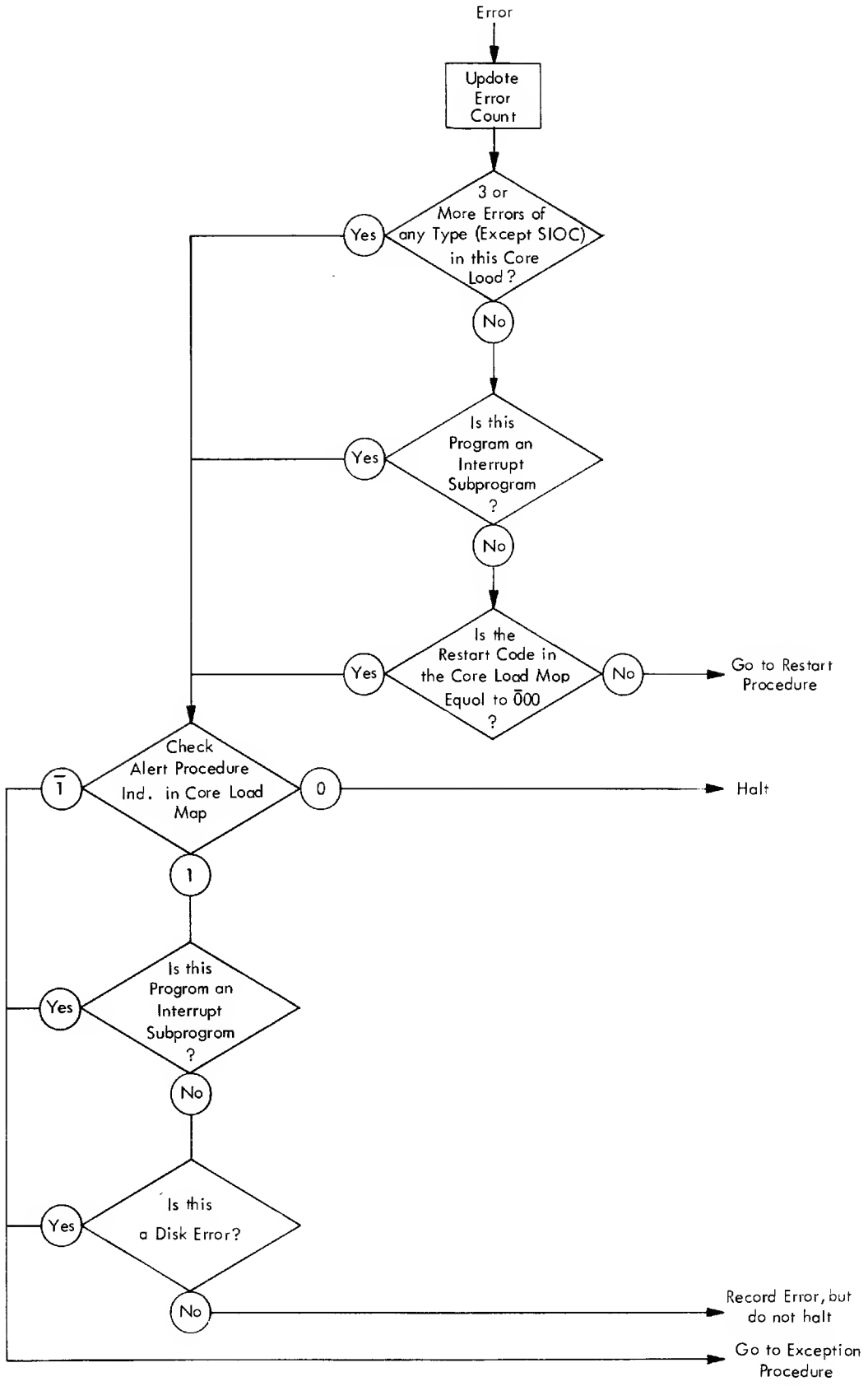


Figure 5. Logic of Corrective Procedure Selection

An IBM Customer Engineer will be able to reinstate the unit on-line by keying in a special code at the console typewriter after depressing the CE Interrupt switch. The Customer Engineer may also disconnect any SIOC unit by keying in a special code at the console typewriter.

Besides the error counters mentioned previously, another counter is used to record errors; this one, however, records errors of any type (except SIOC) and is reset at the end of each core load. It is used by SAC to help determine corrective procedures.

An error message is typed for all error conditions except a disk, TAS, or SIOC instruction that was in error, but was re-executed successfully.

Process Schedule Control (PSC) Program

The main task of the PSC program is to carry out the user's stipulations with regard to core load scheduling. In addition, it performs the duties of servicing recorded interrupts, restarting programs because of error conditions, reading the time clock, and in general, keeping track of the status of core storage at all times.

CALL Statement

At any time during the execution of the mainline program, the user may call PSC to perform any of its aforementioned tasks. The CALL statement and parameter are shown below:

CALL EXEC (IDENT, P_2)

IDENT — 01000 PSC Identification Code (option A)
01001 PSC Identification Code (option B)
01002 PSC Identification Code (option C)
01003 PSC Identification Code (option D)
01004 PSC Identification Code (option E)

P_2 — 00XXX, where XXX is the program identification code for the next mainline program to be executed. P_2 is required only for options B and C.

Option A The next core load will be the one specified in the Core Load Map (see Figure 3).

Option B The next core load will be the one identified by XXX.

Option C The next core load will be the one identified by XXX, which is a special purpose program. (This option differs from Option B in that the user can return to the "calling program" by utilizing Option D.)

Option D Return to the mainline program that called the special purpose program.

Option E Service all recorded interrupts and return to the next statement in the main program.

Scheduling and Loading

The Process Schedule Control program performs its scheduling activities through the use of the Core Load Map (CLM). When a new program is to be brought into core storage, PSC first checks the CALL statement IDENT code and the list of parameters for instructions. If the code specifies that the user's predetermined sequence of programs (per CLM) should be followed, PSC searches the current Core Load table (located in the Skeleton Executive) for the three-digit identifier of the next core load to be loaded. The CLM (on disk storage) is then searched until the record corresponding to this identifier is found. The record is picked out of the CLM to serve as the control and address information for the new program.

If the IDENT code specifies a program other than the one which would be next in sequence, PSC will go directly to the CLM on disk storage and begin searching for the proper record which, when found, will be used for control and address information for the new program.

Using the new map information, PSC then supervises the loading of the new mainline program.

Servicing Recorded Interrupts

As stated previously, the recording of interrupts is a user controlled action; similarly, the servicing of the recorded interrupts is at the user's option. At any time during the execution of the mainline program, the user can call PSC to service all recorded interrupts. If no call is given for this purpose, then PSC, when called at the end of each mainline program, will service all recorded interrupts. In either case, the recorded interrupts will be serviced in the *interruptible* mode.

Serial Input/Output Control (SIOC) Program

The SIOC program directs all input and output operations relative to the Serial Input/Output Channel. By the use of a CALL statement for the SIOC program, data can be read from a manual entry unit or a sense switch unit, or can be written on a digital display unit or an output printer. An operational function of the SIOC program is to put printer messages in a format if specified by the user.

CALL Statement

The SIOC program may be called from any user-written program. The CALL statements and parameters are shown below:

CALL EXEC (IDENT, P_2, P_3)

or

CALL EXEC (IDENT, P_2, P_3, P_4, P_5)

The longer CALL statement from either a mainline program or an interrupt subprogram pertains to the

formatted output data for an output printer. Nonformat data for the output printer or another type of unit will use the shorter CALL statement.

The operands of the above CALL statements are defined as follows:

IDENT – 00600

P_2 – EEMUU, where EE are the control digits which designate the type of device and must be one of the following:

- 10 – output printer
- 11 – digital display unit
- 12 – sense switch unit
- 13 – manual entry unit

M is the modifier to designate the type of operation and must be one of the following:

- 0 – Print a message in a format specified by the user, but actually prepared by the SIOC program.
- 1 – If an output printer is the device selected (control digits = 10), print a message exactly as the user has prepared it in his program; if a manual entry unit is selected (control digits = 13), execute a write operation to turn on the Enter light on the unit; if a digital display unit is selected (control digits = 11), display the message on the unit as the user has laid it out in core storage.
- 2 – Read the designated sense switch or manual entry unit in the *numeric* mode.
- 3 – Read the designated sense switch or manual entry unit in the *alphameric* mode.

And UU is the unit indicator; a two digit constant which indicates the specific unit the user wishes to operate. The constant is identical to the last two digits of the unit indicator associated with the unit being addressed. For example, if 6070 is the unit indicator for the unit, then the user would define UU as 70 in the parameter of the CALL statement.

Short Call

P_3 – The core address of the leftmost position of the message or read-in area.

Long Call

P_3 – Is the statement number of the FORMAT statement to be used in conjunction with the data.

P_4 – Is a literal describing the number of data addresses in the LIST, which follows.

P_5 – The list of the address(es) of data to be transmitted to the output printer. The LIST and the FORMAT statements are used as described in the IBM *1620 Monitor I System Reference Manual* (Form C26-5739).

Input Operations

The SIOC program takes control when one of the following two situations occurs:

1. A CALL statement is executed which specifies either a sense switch unit or a manual entry unit.
2. An interrupt is initiated by the depression of the Execute button on any of the input units.

CALL Statement Procedure. Input devices are read in a masked mode, starting with the lowest numerical address associated with the selected device. After each reading, the address just read is incremented by one until the twelve addresses of a manual entry unit or the four addresses of a sense switch unit have been read.

The data is read into core storage, starting with the address specified by "read-in area" in the CALL statement, and continuing into successively higher core storage locations. If the data to be read is in alphameric form, "read-in area" must specify an odd core location.

Interrupt Procedure. The operator can initiate the reading of an input device by depressing the Execute button on the unit. This causes the user's program to be interrupted, thereby bringing the SIOC program into use. The SIOC program branches to a user's interrupt subprogram which can handle the interrupt in any manner the user desires. There may be one interrupt subprogram for all units or a separate subprogram for each unit. If the user wants to read a unit, a CALL statement must be executed. The operation that ensues is similar to that described under CALL STATEMENT PROCEDURE.

Output Operations

Three of the four available types of devices can be involved in an output operation:

1. Manual Entry Units
2. Digital Display Units
3. Output Printers

Manual Entry Unit. Although a manual entry unit is essentially an input device, it can sometimes be considered an output device. For example, a user's program may require some input data from a manual entry unit. To signal the operator that the data is needed, a CALL statement with a control code of 11 and a modifier of 0 is executed. This causes the SIOC program to turn on the Enter light on the selected device and branch back to the calling program. When the operator has entered the data, he can initiate an interrupt by depressing the Execute button.

Digital Display Unit. When a CALL statement specifies a digital display unit, the SIOC program immediately writes four digits on the unit, starting at the address associated with the "thousands" position of the unit. This position must be addressed first because it resets the unit.

The data to be written must be stored in X through X + 3, where X is the address specified by the P_3 address in the CALL statement.

The "sign" of the four-digit value is transmitted to the sign position in the unit.

Output Printer. The SIOC program section that pertains to the output printers may reside in core with the Skeleton Executive program. This allows the SIOC pro-

gram to return to the user's programs between characters of the message being sent to an output printer.

When the user writes a CALL statement to print a message, he must provide the SIOC program with address information concerning the message. This can be done in two ways: if the message is to be printed exactly as it appears in the program, the user must give the SIOC program the starting address of the message; if the message is to be printed after the SIOC program has placed it in a user-specified format, the user must provide the statement number of the FORMAT statement and the starting address of the output message. If the first method is used, the message data must be in consecutive core storage locations; if the second method is chosen, the message can be composed of data located in different areas of core storage.

Printing the Message. When the SIOC program determines that the message is in the desired format, the message is transmitted to a "disk buffer area." This area, specified by the user at assembly time, can be up to five cylinders in length.

If no previous messages are waiting in the disk buffer area and the SIOC is not busy, the first character of the message is printed immediately and control is returned to the calling program. If some previous message is in the process of being printed, this new message is stored in the buffer area "behind" all previous messages. If the buffer area is full when a new message arrives, the SIOC program "interlocks" and prints messages continuously until there is room for the latest message.

The user must provide a 100-digit buffer area in core storage from which characters can be printed. When the first 100 characters of a message have been written, the next 100 characters are brought into the internal buffer area from the disk buffer area. Upon completion of the message, the SIOC program checks the disk buffer area and starts outputting a new message if one is found to be waiting.

While the output printer is actually operating on a character, the user's mainline program or interrupt subroutine is being executed. When the printer has completed the indicated operation, a signal is generated to initiate an SIOC interrupt which permits the next character to be transmitted to the printer.

Output Printer Programming Considerations. The SIOC program assumes that all messages begin in the numerical mode. If the user wants to start in the alphameric mode when he is formatting his own message, he must place a "change mode" control character (M) at the beginning of the message.

The user is responsible for form feed control. If the printed data is to be spaced properly on form paper, the user must insert the form feed control code (F) at the proper place in the message.

When the first character in a message is Print Red (A), the SIOC program immediately executes a Return Carriage (R) instruction and permits the message from core storage (alert messages are never put on disk storage). If a previous message is in the process of being printed, it is interrupted so that the alert message can be printed. After the SIOC program has completed printing the alert message, it executes a Return Carriage, a Print Black (B), and then continues with the interrupted message.

When a code (A), Print Red, is used *within* a message, it causes the printer to start printing red. Black printing is not resumed until a Print Black code (B) is executed.

Executive Subprograms

The following descriptions cover the subprograms that are supplied to provide the user with specific 1710 operations.

Contact Sense Subprogram

The Contact Sense subprogram is used to read the status of HSCS contacts into core storage.

CALL STATEMENT

CALL CS (P_1, P_2)

where P_1 is a fixed-point variable containing the terminal address as shown in Table 2. P_2 specifies the leftmost position of the field into which the contact status information will be read.

Contact Operate Subprogram

This subprogram will cause the specified point to be closed for 50 ms.

CALL STATEMENT

CALL CO (P_1)

where P_1 is the Terminal Address of the contact to be operated.

Real-Time Clock Subprogram

This subprogram reads the 1711 real-time clock.

Table 2. Contact Sense Points

Terminal Address (Q_{10} and Q_{11} of SLCB Instructions)	Addresses of Points Scanned
00	000–199
01	020–199
02	040–199
03	060–199
04	080–199
05	100–199
06	120–199
07	140–199
08	160–199
09	180–199
10	200–399
11	220–399
12	240–399
13	260–399
14	280–399
15	300–399
16	320–399
17	340–399
18	360–399
19	380–399

CALL STATEMENT

CALL CLOCK (P_1)

where P_1 is the core storage address for the value of the clock to be stored in.

Mask Subprogram

The MASK subprogram allows a program which is being executed in an interruptible mode to be "masked" from all interrupts. Masked interrupts are not lost – they are merely detained until the UNMASK subprogram is called. The Mask indicator (26) is turned on by execution of the MASK subprogram.

CALL STATEMENT

CALL MASK

Unmask Subprogram

The UNMASK subprogram is used to "unmask" the 1710 interrupt feature. Execution of the UNMASK subprogram places the computer in the interruptible mode if the noninterruptible mode exists as a result of the MASK subprogram. The Mask indicator (26) is turned off by execution of the UNMASK subprogram.

CALL STATEMENT

CALL UNMASK

Manual Entry Subprogram

This subprogram reads in seven digits of information from the 1711 Manual Entry Switches.

CALL STATEMENT

CALL MEOP (P_1, P_2)

where P_1 is the core storage address where the data of the two high-order Manual Entry switch positions is to be stored, and P_2 is the core storage address where the data of the five low-order Manual Entry switches is to be stored.

Analog to Digital Conversion Subprogram

This subprogram causes the analog signal specified to be read into the ADC register for conversion. The converted signal is later read into the computer by execution of the RDADC subprogram.

CALL STATEMENT

CALL ADC (P_1)

where P_1 is the terminal address of the signal to be converted.

Read ADC Register Subprogram

This subprogram reads the contents of the ADC register into core storage and causes another analog signal to read into the ADC register for conversion.

CALL STATEMENT

CALL RDADC (P_1, P_2)

where P_1 is the terminal address of the *next* signal to be converted and P_2 is the core storage address where the present contents of the ADC register are to be stored.

The ADC and RDADC subprograms may be used together to read a series of points. Assume ITAS is a dimensioned variable containing a list of addresses of analog input points and IDRO is a dimensioned variable where the converted signal of each analog input point is to be stored. The following statements would cause points corresponding to ITAS 6 through ITAS 25 to be read, converted, and stored in IDRO 6 through IDRO 25 .

```
      CALL ADC (ITAS (6))
      DO 20 I = 6, 25
   10 CALL RDADC (ITAS (I + 1), IDRO (I))
   20 CONTINUE
```

When RDADC is called, the program is interlocked until conversion of the previous signal is completed. The total conversion time is 50 milliseconds. Statements to perform diagnostic analysis and conversion to engineering units of the readings may be inserted between statement numbers 10 and 20 to overlap this conversion time.

IBM 1710 Printed in U.S.A. C26-5733-0

IBM
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York